How To...
Master Data Governance for Financials:
Send a Mail notification during the governance process

Applicable Releases:
EhP5

Version 1.0
August 2011
## Document History

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>First official release of this guide</td>
</tr>
</tbody>
</table>
## Typographic Conventions

<table>
<thead>
<tr>
<th>Type Style</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example Text</strong></td>
<td>Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Cross-references to other documentation</td>
</tr>
<tr>
<td><strong>Example text</strong></td>
<td>Emphasized words or phrases in body text, graphic titles, and table titles.</td>
</tr>
<tr>
<td><strong>Example text</strong></td>
<td>File and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.</td>
</tr>
<tr>
<td><strong>Example text</strong></td>
<td>User entry texts. These are words or characters that you enter in the system exactly as they appear in the documentation.</td>
</tr>
<tr>
<td><code>&lt;Example text&gt;</code></td>
<td>Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.</td>
</tr>
<tr>
<td><strong>EXAMPLE TEXT</strong></td>
<td>Keys on the keyboard, for example, F2 or ENTER.</td>
</tr>
</tbody>
</table>

## Icons

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Caution Icon]</td>
<td>Caution</td>
</tr>
<tr>
<td>![Note or Important Icon]</td>
<td>Note or Important</td>
</tr>
<tr>
<td>![Example Icon]</td>
<td>Example</td>
</tr>
<tr>
<td>![Recommendation or Tip Icon]</td>
<td>Recommendation or Tip</td>
</tr>
</tbody>
</table>
Table of Contents

1. Business Scenario ........................................................................................................... 1
2. Background Information ................................................................................................. 1
3. Step-by-Step Procedure ................................................................................................... 2
   3.1 Create a New Workflow ............................................................................................... 2
   3.2 Modify the Workflow Container .................................................................................. 4
   3.3 Modify Activation Step ............................................................................................... 7
   3.4 Add a Mail Notification Step ..................................................................................... 11
   3.5 Configuration ............................................................................................................ 13
   3.6 Example Coding for Program-Exit Class ................................................................. 14
1. **Business Scenario**

SAP Master Data Governance for Financials (MDGF) provides business processes to find, create, change and delete financial master data. It supports the governance of financial master data in a central hub and the distribution to connected operational and business intelligence systems.

Master data in MDGF is maintained by change request. The processes of MDGF are business workflow-driven and can include several approval and revision phases, and the collaboration of all users participating in the master data maintenance.

This scenario addresses when a change request is finally approved, all users affected will get an Email with a protocol of the change request. The Email contains change request metadata as well as the data that has been maintained during processing of the change request.

This How To Guide describes possible solution to send out Email notification from the workflow.

2. **Background Information**

A new workflow step is added into the workflow template. You may adjust at which point the sending mail step can be inserted as long as you have maintained the elements in workflow container.
3. Step-by-Step Procedure

The following is a description of how to send an Email notification after activating a change request. The scenario takes workflow WS75700040 as an example. The workflow WS75700040 only contains three dialog steps.

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>WS75700040</td>
<td>1</td>
<td>Processing</td>
</tr>
<tr>
<td>WS75700040</td>
<td>2</td>
<td>Final Check</td>
</tr>
<tr>
<td>WS75700040</td>
<td>3</td>
<td>Revision</td>
</tr>
</tbody>
</table>

The idea is that add a program-exit to the activation step of workflow. After the activation step add a new step with type “Send Email”.

Example is developed in QV5/Client 005, tested in QV5/Client 405.

3.1 Create a New Workflow

1. At cross client, enter transaction SWDB. Fill workflow “WS75700040” and press ENTER

2. Select Workflow in menu and select “Save as…”
3. In the prompted window, fill in abbreviation and name. Click.

4. A new workflow number is generated automatically. Activate the workflow.
3.2 Modify the Workflow Container

1. Create a new structure and new table type in the DDIC using transaction SE11. This will handle the assigned object list of the change request.
   a. Create a data type "Structure"
      
      ![Structure creation interface]

   b. Create a data type "Table Type"
2. Go to transaction SWDD, input the workflow that just created automatically. Select “Workflow Container” at the left side.

3. Use the context menu to create two new container elements.
a. EMAILRECEIVER

This element is used to store users that need to be notified by Email. It's of type SWHACTOR. The properties are shown as below.

b. CREQUEST_OBJLST

This element is used to store master data in change request.
3.3 Modify Activation Step

1. In transaction SWDD, select step “000298 Änderungen aktivieren” in Navigation area.
Send a Mail Notification During the MDGF Process

2. At right panel, choose the option “Copy and Edit Task” from the dropdown list of Task ID.

3. Fill in all fields like following.

   Note: There may be errors in setting work item text here. You may set the work item text after binding.
4. In tab Container, create two new container elements CREQUEST_OBJLST and EMAILRECEIVER. Set the data type and properties just like before. Save the task and go back to the step definition.

5. Binding the workflow container element with task container element.
6. Create a new class using transaction SE24, which inherits interface IF_SWF_IFS_WORKITEM_EXIT. Implement the method IF_SWF_IFS_WORKITEM_EXIT->EVENT_RAISED to set the container element EMAILRECEIVER and CREQUEST_OBJLST. The example class ZCL_WORKFLOW_EXIT is added at bottom.

7. In step activation, go to tab Program Exits and add class ZCL_WORKFLOW_EXIT. The example code of the class can be found in 3.6 Example Coding for Program-Exit Class.

8. Save the workflow and click ✔ at the top of the activity.
3.4 Add a Mail Notification Step

1. At the left area of the workflow builder, select „Step Types That Can Be Inserted”, and then move the „Send Mail” and drop down after the activation step.

![Workflow Builder Screenshot](image)
2. Fill in information to the step. You may insert element via icon "Insert Expression".

3. Click tab Control and create a new task.

4. Double click the new task and create a new element CREQUEST_OBJLIST in the container just like steps before. If you run into locking issues, maintain the standard task using transaction PFTS.

5. Binding between workflow container element and task container element. (Availability of the parameters SIGN & ENCRYPT depend on the setup of the workflow and might not be available.)
Send a Mail Notification During the MDGF Process

September 2011

1. At MDG client, go to transaction MDGIMG. Go to node General Settings->Process Modeling->Workflow->Other MDG Workflows->Define Workflow Step Numbers. Add three steps for the new created workflow.

2. Create a change request type with the new created workflow. The change request type can be created at MDGIMG->General Settings->Process Modeling->Change Requests->Create Change Request Type

3. Assign agents in MDGIMG->General Settings->Process Modeling->Workflow->Other MDG Workflows->Assign processor to Workflow Step Number (Simple Workflow)

4. To make users in SAP system can receive and email, following settings are needed.
   a. Add the email address in SU01 (or make sure it’s there)
   b. Set „Comm. Method“ to „E-mail“ in SU01
   c. In SO16, tab Mail sy.grp, radio button „Send to Home Addresses of Users"
   d. In SO16, tab Addressing, Reset buffer

6. Save the workflow and click ✔️ at the top of the activity.

7. Activate the whole workflow.

3.5 Configuration
### Example Coding for Program-Exit Class

#### Example Coding:

```plaintext
METHOD if_swf_ifs_workitem_exit~event_raised.
  DATA: lo_workitem TYPE REF TO if_swf_ifs_parameter_container,
       lt_objlist TYPE usmd_ts_crequest_objkey,
       lv_return TYPE sysubrc,
       ls_swotobjid TYPE swotobjid,
       lv_crequest TYPE usmd_crequest,
       lv_edition TYPE usmd_edition,
       lv_edtn_type TYPE usmd_edition_type,
       lv_model TYPE usmd_model,
       lo_model TYPE REF TO if_usmd_model,
       lt_message TYPE usmd_t_message,
       lt_message1 TYPE usmd_t_message,
       lt_objlist_ele TYPE zusmd_t_entity,
       lr_data TYPE REF TO data,
       ls_objlist TYPE usmd_s_crequest_objkey,
       ls_objlist_ele TYPE zusmd_s_entity,
       lv_obj TYPE string,
       ls_entity_prop TYPE usmd_s_entity_prop,
       ls_fdep TYPE usmd_s_fdep,
       ld_entity TYPE usmd_entity,
       lv_entity_txt TYPE usmd_value,
       lv_field_txt TYPE usmd_value,
       ls_receiver TYPE swhactor,
       lt_receiver_temp TYPE tswhactor,
       lt_receiver TYPE tswhactor,
       lr_badi_wf_agent TYPE REF TO usmd_wf_agent,
       ld_agent_filter TYPE swcont-value,
       lo_table TYPE REF TO cl_abap_tabledescr,
       lo_struct TYPE REF TO cl_abap_structdescr,
       lo_line TYPE REF TO cl_abap_datadescr,
```

---

Send a Mail Notification During the MDGF Process

**3.6 Example Coding for Program-Exit Class**

<table>
<thead>
<tr>
<th>Ty.</th>
<th>Parameter</th>
<th>Type spec.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>EVENT_NAME</td>
<td><code>SW_WK_EVENT_TYP</code></td>
<td>Workflow Event Type for Workflow Runtime</td>
</tr>
<tr>
<td>IN</td>
<td>WORKITEM_Context</td>
<td>TYPE REF TO <code>IF_WFK_WORKITEM_CONTEXT</code></td>
<td>Context for Work Item</td>
</tr>
<tr>
<td>IN</td>
<td>IF_IWFS_WORKITEM_EXIT_ERROR</td>
<td></td>
<td>Programming Exit Exception</td>
</tr>
</tbody>
</table>

**Class Builder: Class ZCL_WORKFLOW_EXIT Display**

- **METHOD** if_swf_ifs_workitem_exit~event_raised.
- **DATA:**
  - `lo_workitem` TYPE REF TO if_swf_ifs_parameter_container,
  - `lt_objlist` TYPE usmd_ts_crequest_objkey,
  - `lv_return` TYPE sysubrc,
  - `ls_swotobjid` TYPE swotobjid,
  - `lv_crequest` TYPE usmd_crequest,
  - `lv_edition` TYPE usmd_edition,
  - `lv_edtn_type` TYPE usmd_edition_type,
  - `lv_model` TYPE usmd_model,
  - `lo_model` TYPE REF TO if_usmd_model,
  - `lt_message` TYPE usmd_t_message,
  - `lt_message1` TYPE usmd_t_message,
  - `lt_objlist_ele` TYPE zusmd_t_entity,
  - `lr_data` TYPE REF TO data,
  - `ls_objlist` TYPE usmd_s_crequest_objkey,
  - `ls_objlist_ele` TYPE zusmd_s_entity,
  - `lv_obj` TYPE string,
  - `ls_entity_prop` TYPE usmd_s_entity_prop,
  - `ls_fdep` TYPE usmd_s_fdep,
  - `ld_entity` TYPE usmd_entity,
  - `lv_entity_txt` TYPE usmd_value,
  - `lv_field_txt` TYPE usmd_value,
  - `ls_receiver` TYPE swhactor,
  - `lt_receiver_temp` TYPE tswhactor,
  - `lt_receiver` TYPE tswhactor,
  - `lr_badi_wf_agent` TYPE REF TO usmd_wf_agent,
  - `ld_agent_filter` TYPE swcont-value,
  - `lo_table` TYPE REF TO cl_abap_tabledescr,
  - `lo_struct` TYPE REF TO cl_abap_structdescr,
  - `lo_line` TYPE REF TO cl_abap_datadescr,
Send a Mail Notification During the MDGF Process

September 2011

```
ls_comp           TYPE abap_compdescr,
lo_type           TYPE REF TO cl_abap_typedescr.

FIELD-SYMBOLS:  <f_obj>           TYPE any,
                 <f_value>         TYPE any,
                 <f_struct>        TYPE any,
                 <f_obj_tab>       TYPE ANY TABLE.

* The email is sent only after the activation of master data
  CHECK im_event_name EQ if_swf_ifs_workitem_exit=>c_evttyp_after_execution.

*Get instance of workflow item
  lo_workitem = im_workitem_context->get_wi_container( ).

* Get Change request number
  TRY.
    CALL METHOD lo_workitem->get
       EXPORTING
       name   = '_WI_OBJECT_ID'
       IMPORTING
       value  = ls_swotobjid.
    lv_crequest = ls_swotobjid->objkey+0(12).
    CATCH cx_swf_cnt_elem_not_found .
    CATCH cx_swf_cnt_elem_type_conflict .
    CATCH cx_swf_cnt_unit_type_conflict .
    CATCH cx_swf_cnt_container .
  ENDTRY.

* Get object list of the change request
  * Get data model
    SELECT SINGLE usmd_edition FROM usmd120c INTO lv_edition WHERE usmd_crequest = lv_crequest.
    IF sy-subrc EQ 0.
      IF sy-subrc EQ 0.
        SELECT SINGLE usmd_model FROM usmd010c INTO lv_model WHERE usmd_edtn_type = lv_edtn_type.
        ENDIF.
      ENDIF.
    ENDIF.

    IF lv_model IS NOT INITIAL.
      CALL METHOD cl_usmd_model=>get_instance
         EXPORTING
         i_usmd_model = lv_model
         IMPORTING
         eo_instance  = lo_model
         et_message   = lt_message.
  ```
ENDIF.

IF lt_message IS INITIAL.
  CALL METHOD lo_model->if_usmd_model_crequest~crequest_get_objlist
  EXPORTING
    i_crequest = lv_crequest
  IMPORTING
    ets_crequest_objkey = lt_objlst
    et_message = lt_message1.
ENDIF.

APPEND LINES OF lt_message1 TO lt_message.

LOOP AT lt_objlst INTO ls_objlst.
  CLEAR ls_objlst_ele.
  ls_objlst_ele-entity = ls_objlst-fieldname.
  ASSIGN ls_objlst-r_data->* TO <f_obj>.
  lo_type = cl_abap_typedescr->describe_by_data(<f_obj>).
  CHECK lo_type->kind = cl_abap_typedescr->kind_table.
  lo_table ?= lo_type.
  lo_line = lo_table->get_table_line_type( ).
  ASSIGN <f_obj> TO <f_obj>.
  IF lo_line->kind = cl_abap_typedescr->kind_struct.
    lo_struct ?= lo_line.
    IF lo_line->kind = cl_abap_typedescr->kind_struct.
      lo_line = lo_line->get_table_line_type( ).
      ASSIGN <f_obj> TO <f_obj>.
      IF <f_obj> IS ASSIGNED.
        CLEAR lv_obj,ld_entity.
        * Get text of entity type
        CALL METHOD cl_usmd_services->fieldname2entity
          EXPORTING
            io_model = lo_model
            i_fieldname = ls_objlst-fieldname
          RECEIVING
            r_entity = ld_entity.

        CALL METHOD cl_usmd_services->entity2text
          EXPORTING
            io_model = lo_model
            i_entity = ld_entity
          RECEIVING
            r_text = lv_entity_txt.
        CONCATENATE lv_entity_txt ':' INTO lv_obj SEPARATED BY space.

        * Get master key of ld_entity
        READ TABLE lo_model->dt_entity_prop INTO ls_entity_prop WITH KEY usmd_entity = ld_entity.
LOOP AT lo_model-
>dt_fdep INTO ls_fdep WHERE fieldname EQ ls_entity_prop-r_fprop-
>fieldname.
  IF sy-subrc EQ 0.
    ASSIGN COMPONENT ls_fdep-
masterfield OF STRUCTURE <f_struct> TO <f_value>.
    ASSERT <f_value> IS ASSIGNED.
  ENDIF.
* Get text of field name
  CALL METHOD cl_usmd_services=>fieldname2text
    EXPORTING
      io_model = lo_model
      i_fieldname = ls_fdep-masterfield
    RECEIVING
      r_value = lv_field_txt.
  CONCATENATE lv_obj lv_field_txt ' ' <f_value> ' ' INTO lv_obj SEPARATED BY space.
ENDLOOP.
  ls_objlst_ele-value = lv_obj.
  APPEND ls_objlst_ele TO lt_objlst_ele.
ENDLOOP.
ENDIF.
ENDIF.
ENDLOOP.

TRY.
  CALL METHOD lo_workitem->set
    EXPORTING
      name = 'CREQUEST_OBJLST'
      value = lt_objlst_ele.
  CATCH cx_swf_cnt_elem_not_found.
  CATCH cx_swf_cnt_elem_type_conflict.
  CATCH cx_swf_cnt_unit_type_conflict.
  CATCH cx_swf_cnt_container.
ENDTRY.

****************************************
*********************
****
** Set parameter EMAILRECEIVER
** Call BAdI to get agents
  IF ld_agent_filter IS INITIAL.
    ld_agent_filter = if_usmd_wf_agent=>gc_filter- standard.
  ENDIF.

  GET BADI lr_badi_wf_agent
    FILTERS
      agent_filter = ld_agent_filter.
  IF lr_badi_wf_agent IS BOUND.
    CALL BADI lr_badi_wf_agent->get_agents
      EXPORTING

id_appstep  = '1'
id_crequest  = lv_crequest
if_single_proc  = '
?id_index  = 0

IMPORTING
  et_agent  = lt_receiver_temp
  et_message  = lt_message.
ENDIF.
APPEND LINES OF lt_receiver_temp TO lt_receiver.

IF lr_badi_wf_agent IS BOUND.
  CALL BADI lr_badi_wf_agent->get_agents
  EXPORTING
    id_appstep  = '2'
id_crequest  = lv_crequest
if_single_proc  = '
?id_index  = 0

IMPORTING
  et_agent  = lt_receiver_temp
  et_message  = lt_message.
ENDIF.
APPEND LINES OF lt_receiver_temp TO lt_receiver.

IF lr_badi_wf_agent IS BOUND.
  CALL BADI lr_badi_wf_agent->get_agents
  EXPORTING
    id_appstep  = '3'
id_crequest  = lv_crequest
if_single_proc  = '
?id_index  = 0

IMPORTING
  et_agent  = lt_receiver_temp
  et_message  = lt_message.
ENDIF.
APPEND LINES OF lt_receiver_temp TO lt_receiver.

TRY.
  CALL METHOD lo_workitem->set
  EXPORTING
    name  = 'EMAILRECEIVER'
    value  = lt_receiver.
  CATCH cx_swf_cntElemNot Found.
  CATCH cx_swf_cntElemTypeConflict.
  CATCH cx_swf_cntUnitTypeConflict.
  CATCH cx_swf_cntContainer.
ENDTRY.

IF lv_return IS INITIAL.
ENDIF.
ENDMETHOD.
www.sdn.sap.com/irj/sdn/howtoguides